2014 47th Annual IEEE/ACM International Symposium on Microarchitecture

A Practical Methodology for Measuring the Side-Channel Signal Available to the Attacker for Instruction-Level Events

Robert Callan, Alenka Zajić, and Milos Prvulovic
Georgia Institute of Technology

***Abstract*—This paper presents a new metric, which we call Signal Available to Attacker (SAVAT), that measures the side channel signal created by a specific single-instruction difference in program execution, i.e. the amount of signal made available to a potential attacker who wishes to decide whether the program has executed instruction/event A or instruction/event B. We also devise a practical methodology for measuring SAVAT in real systems using only user-level access permissions and common measurement equipment. Finally, we perform a case study where we measure electromagnetic (EM) emanations SAVAT among 11 different instructions for three different laptop systems. Our findings from these experiments confirm key intuitive expectations, e.g. that SAVAT between on-chip instructions and off-chip memory accesses tends to be higher than between two on-chip instructions. However, we find that particular instructions, such as integer divide, have much higher SAVAT than other instructions in the same general category (integer arithmetic), and that last-level-cache hits and misses have similar (high) SAVAT. Overall, we confirm that our new metric and methodology can help discover the most vulnerable aspects of a processor architecture or a program, and thus inform decision-making about how to best manage the overall side channel vulnerability of a processor, a program, or a system.**

I. Introduction

Side channels are a powerful class of attacks that circumvent traditional security protections and access controls. Unlike traditional attacks that exploit vulnerabilities in what the system does, side channel attacks allow information to be obtained by observing how the system does it.

An example of a side channel attack is illustrated in Figure 1. There Alice, an investment banker, makes several stock trades and writes sensitive emails using an encrypted connection to her employer's server. Alice's back is against the wall because she does not want others to observe what she is doing. However, this coffee shop is known to be frequently visited by investment bankers, so at the next table Eve is recording electromagnetic (EM) emanations from Alice's laptop using an antenna hidden in a briefcase, Evan has installed a microphone under the table frequented by Alice to collect sound emanations from her laptop, and Evita has attached a power meter, disguised as a battery charger, into the wall socket where Alice's laptop is plugged in.

Computation requires a large variety of electronic and microarchitectural activity. Figure 1 only illustrates attacks that observe power [6], [25], [34], [43], acoustic noise [4], [12], [51], and EM emanations [2], [32]. Other side channel attacks may monitor behavior under faults [8], [24], detect when cache misses occur [5], [61], [63], analyze behavior of instruction caches and branch predictors [1], etc. In general, side channel attacks are carried out by 1) identifying some physical or microarchitectural "signal" that "leaks" desired information about system activity or the data it processes, and then 2) monitoring and analyzing that signal as the system operates. Much work has been done to prevent particular side channel attacks, either by severing the tie between sensitive information and the side channel signal, or by trying to make the signal more difficult to measure. However, such work mostly focuses on preventing a particular side channel attack in a very specific piece of code, such as a cryptographic kernel.

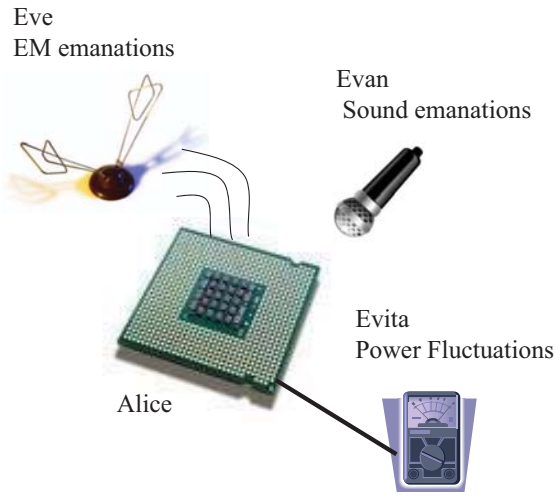

Figure 1. Examples of side channel attacks.

Strategies for quantifying potential side channel exposure *at the microarchitectural and architectural levels* are still an open problem. The current state of the art is the recently proposed Side-Channel Vulnerability Factor (SVF) [15], [16] which measures how the side channel signal correlates with high-level execution patterns (e.g. program phase transitions). While this metric allows overall assessment of the "leakiness" of a particular system and application over a given side channel, it provides limited insight to 1) computer architects about which architectural and microarchitectural features are the strongest leakers, and to 2) software developers about how to reduce the side channel leakiness of their code.

This paper is the first step toward such fine-grain under-

1072-4451/14 $31.00 © 2014 IEEE
DOI 10.1109/MICRO.2014.39

standing and assessment. Our main contributions are:

1) A new metric, Signal Available to Attacker (SAVAT), that quantifies the side channel signal caused by differences in code execution,
2) A practical methodology for measuring SAVAT on real machines, and
3) A case study that defines SAVAT for an EM emanations side channel and a small set of instructions. SAVAT is measured on several different machines and analyzed to show how these results might be used by computer architects and software developers.

The basic premise of our work is that information leakage through side channels occurs when the instructions executed depend on sensitive information, and this instruction-level difference creates a side channel signal that is available to the attacker. Thus the difficulty with which an attacker can obtain the secret information depends on both 1) program activity: the information-dependent difference created at the instruction level, and 2) the side channel signal's dependence on these instruction-level differences. Our SAVAT metric quantifies this second property for a system, allowing 1) programmers to change their code to avoid creating high-SAVAT instruction-level differences that depend on secret information, and 2) computer architects and microarchitects to focus their side channel mitigation efforts on high-SAVAT instructions.

We also introduce a new methodology for directly measuring SAVAT for a pair of instructions in a system. The goal is to measure the side channel signal due to the differences between two instructions/events (call them A and B). A naïve approach measures the signals for A and B separately, then computes the area (total difference over time) between the signal curves for A and B. Unfortunately, this naïve approach has a very large measurement error. First, the single-instruction signal difference is much smaller than the overall signal generated by the execution that surrounds the instruction under examination. Computing a small difference between two large signals is subject to huge relative error because the measurement error for each signal is proportional to the signal's overall value, i.e. the difference between signals might be dominated by measurement errors in the two measurements. Second, the computed $A-B$ signal is affected by imperfect alignment of the two signals in time. Third, this approach requires recording many samples of the two signals (to enable accurate subtraction) over a very short period of time (the duration of a single instruction). Even the most sophisticated ($>$\$200,000 cost) instruments provide only 10-20 samples per clock cycle in modern multi-GHz processors.

To solve this problem, our measurement methodology creates a pattern of activity in the system that alternates between the two instructions/events under test and then measures the periodic signal this alternation creates. The measured signal is created by a large number of instruction-pair differences, so it can be measured with much less sophisticated instruments. Furthermore, this method measures the A vs B difference directly (reducing the relative error) and can be performed at much lower frequencies using commercially available instruments.

Finally, we perform a case study where we measure the EM side channel SAVAT for all possible pairings of 11 instructions selected from the x86 instruction set, on three different laptop systems. We demonstrate our methodology on EM side channel emanations because such signals are generally very weak and can be measured non-destructively using measurement instruments available in our lab. The results of the case study confirm the intuitive expectations that 1) off-chip accesses (cache misses that go to main memory) vs on-chip activity have a high SAVAT and that 2) instructions with similar activity (e.g. ADD and SUB) have a very low mutual SAVAT. However, we also find that, for attacks from shorter distances, cache hits in large caches are also easily distinguished from other operations - just as easily as off-chip memory accesses are, and that among arithmetic instructions, execution of an integer divide instruction is by far the easiest to distinguish.

The results of these measurements can be used by circuit designers and microarchitects to reduce susceptibility to side channel attacks by focusing on high-SAVAT aspects of their designs (e.g. off-chip memory accesses, last-level-cache hits, and possibly the integer divider in the systems we measured). Programmers, compilers, and algorithm designers can also use SAVAT to guide code changes to avoid using "loud" activity when operating on sensitive data. Although we only present results for the EM side channel, we believe that our overall methodology can be used to characterize the relationship between instruction-difference and signal-difference in other side channels, especially acoustic and power-consumption side channels where instruments are readily available to measure the power of the periodic signals created by our methodology.

The rest of this paper defines our new metric, Signal Available to Attacker (SAVAT), and explains the rationale behind it (Section II), describes our novel measurement methodology for SAVAT (Section III), describes a measurement setup we used to measure pairwise SAVAT for 11 different instructions/events (Section IV), shows our measurement results (Section V), summarizes related work (Section VI), and presents our conclusions (Section VII).

II. A New Metric: Signal Available to Attacker (SAVAT)

A program that uses some data as its input will generate data-dependent activity in the processor and possibly also in the off-chip memory and other system components. This data-dependent activity will create signals in various side channels. Data-dependent activity in the system cannot be avoided: even if the program's control flow does not depend

on the value of the input, and if the circuitry of the processor is designed such that every operand value results in the exact same overall number of bit-flips in transistors and wires, there will be at least some transistors or wires whose switching activity is input-dependent. This difference in transistor/wire activity creates a difference in various physical side channel signals, such as EM emanations, power consumption, etc. Process variations, physical location of the circuit, etc. allow side channel signals to be created even if the circuitry is designed to minimize the operand-dependent variations in overall activity – these techniques can dramatically reduce the magnitude of data-dependent signal variation but cannot completely eliminate them. But this does not mean that these and other techniques are ineffective – they force attackers to use more expensive, bulkier, and less widely available snooping devices, to run more risk of discovery (e.g. if they get closer to collect the weak EM emanations), and/or to need more data points and collect signals longer for the same amount of extracted information.

Most side channel mitigation techniques are expensive, especially if applied very broadly. For example, circuit-level techniques that mask input-dependent variations in overall activity do so by performing more activity overall: when actual inputs require little activity, additional unnecessary activity is performed to match what happens for high-activity values. This minimizes variations in power consumption, EM activity, etc. The costs of these techniques are high: large increases in chip area (for dummy-activity circuitry), execution times that always match the worst case inputs, and power consumption that always equals the peak power consumption.

To allow judicious use of side channel mitigation techniques, we need a metric that quantifies the amount of side channel signal that is made available to the attacker when there is a particular variation in processor and system activity. To define this metric, we must first choose the granularity for variation in activity. At the coarsest granularity, we can consider execution of different programs. While such a coarse-grain metric would provide some ability to compare different systems in terms of their "leakiness" in a particular side channel for a particular application, this metric does not provide much guidance for where to apply mitigation techniques. The next level of granularity is program phases. Such a phase-granularity metric would be used similarly to SVF [15], [16], i.e. it would allow more robust comparisons of different systems and specific mitigation approaches, but would not provide specific feedback about where mitigation might be needed or what type of mitigation might be appropriate.

At the other extreme, measurement of overall side channel signals created by transistor- and wire-granularity variation in activity might allow some low-level layout optimizations. However, such a metric 1) would be very difficult to measure on real systems – it would require one to somehow trigger activity on that specific transistor or wire while creating little or no activity on all other transistors and wires in the system under test, and 2) would only be useful to those with layout-level knowledge of the system's design, i.e. it would be largely meaningless to programmers and unsuitable for most architecture-level studies.

To allow real-system measurements and provide specific mitigation guidance to programmers and computer architects, the granularity we use is an instruction-level event. Our metric, which we call Signal Available to Attacker (SAVAT), is a measure of the overall signal that is made available to the attacker through the side channel as a result of a single-instruction variation: executing a different instruction because of a control-flow decision, having or not having a cache miss, etc. The SAVAT is a pairwise metric - it measures the signal made available to the attacker when we execute instruction/event A instead of executing instruction/event B (or vice versa). For example, the ADD/MUL SAVAT is the overall amount of the side channel signal that tells the attacker whether we have executed an ADD or a MUL instruction, the LDM/LDL2 SAVAT is the overall amount of the side channel signal that tells the attacker whether we had a L2 hit or an off-chip memory access for a load instruction, etc. We also define the single-instruction SAVAT as the maximum of the pairwise SAVATs where both events in the pair are generated using the same instruction. For example, the SAVAT for a load instruction is the maximum of pairwise LDM/LDM, LDM/LDL2, LDM/LDL1, etc. SAVATs.

III. A Methodology for Measuring SAVAT in Real Systems

To measure the SAVAT for a pair of instructions in a real system, we must force the system to execute those two instructions/events in a way that minimizes the activity variations from all other system activities, and then we must actually measure the resulting side channel signals. A naïve approach for measuring an A/B SAVAT is illustrated in Figure 2: execute a program fragment that performs instruction/event A and record the side channel signal, then execute an identical program fragment but now with instruction/event B instead of A, record the side channel signal again, then align the two signal curves in time and compute the area between the two curves.

This approach suffers from several major problems. First, in many side channels the signal change from a single instruction/event is extremely weak relative to the overall signal – for EM emanations used in our case, the signal energy produced by a single-instruction/event is well below the detection threshold of a high-end measurement instrument. Second, the difference between signals is usually much smaller than the magnitude of these signals, and the measurement error is often relative to the magnitude of the

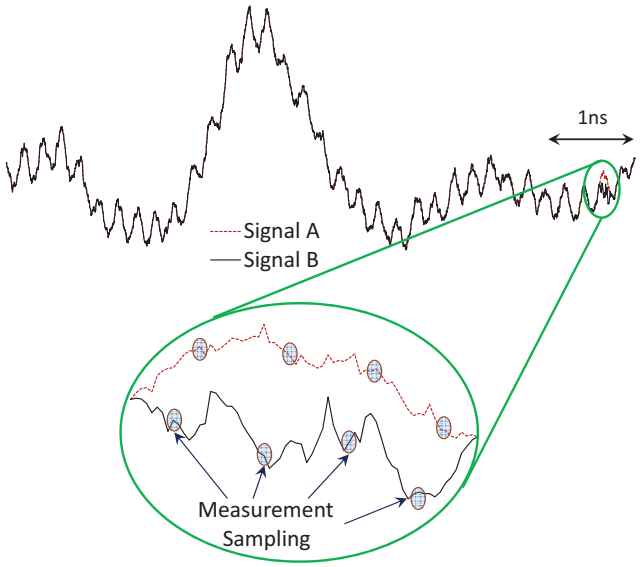

Figure 2. A naïve approach for measuring SAVAT.

measured signal. This means that the relative error in the computed difference can be very large. For signals shown in Figure 2, for example, random measurement error that averages 0.5% of the signal's range will make the two overall curves have a total difference that is >5 times as large as the actual difference (area between the magnified sections of these curves) we are trying to measure. In addition, as illustrated in Figure 2, the execution of instructions/events of interest takes very little time - e.g. the ADD/SUB SAVAT is based on comparing signals that each last only a fraction of a nanosecond, and accurate assessment of their difference requires many samples (at least 10, preferably more) to be collected during that time. This requires instruments with an extremely high real-time sampling rate. Unfortunately, real-time instruments (oscilloscopes or A/D converters) that can digitize signals at rates >50Gsamples/s cost hundreds of thousands of dollars. The few samples that are collected during the "interesting" time period provide a very coarse representation of the signals, and additional error may be produced if the samples in the two measurements are not aligned with each other (as shown in the magnified part of Figure 2). Even when the start of the A and B instructions are aligned, if the A instruction takes longer to complete than the B instruction, a portion of A's execution is compared to unrelated processor activity in the signal containing B. Furthermore, signals are attenuated as they propagate from the source in the processor to the attacker some distance away, and this attenuation is far greater at high frequencies, further complicating the measurement.

To overcome these problems, our measurement methodology proposes directly analyzing the signal created by the execution of code containing both A and B instructions. This code is carefully constructed so that any signal due to differences between the A and B instructions is localized in frequency (Figure 3b), whereas the naïve approach attempts to localize this difference in time in separate A and B signals (Figure 2). The new A and B combined signal is constructed by having the computer system alternate between the two instructions/events (A and B) many times per second as shown in Figure 3b. This alternation generates a periodic signal at the alternation frequency that corresponds to the overall difference between the individual signals. This periodic signal can then be filtered to reject other frequencies including the noise and "uninteresting" signals they carry, and the filtered signal's magnitude can then be measured. For EM emanations, power, sound, etc., this filtering and measurement can be done very precisely using a spectrum analyzer. The spectrum obtained in this way measures the difference in signal strength between A and B instructions/events over a unit time (e.g. a second), and overcomes all of the problems with the naïve measurement because 1) the measured A/B difference signal accumulates over many A/B differences over this one second, effectively amplifying the signal and suppressing noise (the instrument only needs to be sensitive enough to measure the one-second total; we can still compute the single-instruction/event SAVAT by dividing the measured signal by the number of A/B instances that occur each second), 2) the difference between A and B side channel signals is directly measured, without the relative-error problem present when measuring A and B signals separately, and 3) the signal is measured at the alternation frequency, which can be adjusted in software by changing the number of A and B events per iteration of the alternation loop, so we can easily bring that frequency within the measurement range of commercially available instruments. We also have the freedom to select a frequency with relatively little noise - an important consideration for EM emanation side channels where direct collection of A and B side channel signals is subject not only to measurement error but also to noise from various radio signals. Also, while the A/B difference signal occurs at the greatly attenuated high frequencies in the naïve measurement, the A/B difference signal occurs at a single, known, better propagating lower frequency in this new methodology.

This methodology targets an attack model where sensitive information creates differences in execution, which in turn cause different signals in a side channel. Many attacks fit this model. For example, modular exponentiation in RSA is typically implemented in a way that results in testing the bits of the secret exponents one at time, and multiplying two large numbers (e.g. 2048 bits) whenever such a bit is 1. This entire multiplication can thus be viewed as the difference in execution caused by sensitive information (a bit of the exponent). This example also shows that, although the signal leaked by a single-instruction difference can be small, a practical attack may accumulate many of these single-instruction differences – an entire large-numbers multiplication in this example. How many single-instruction differences need to

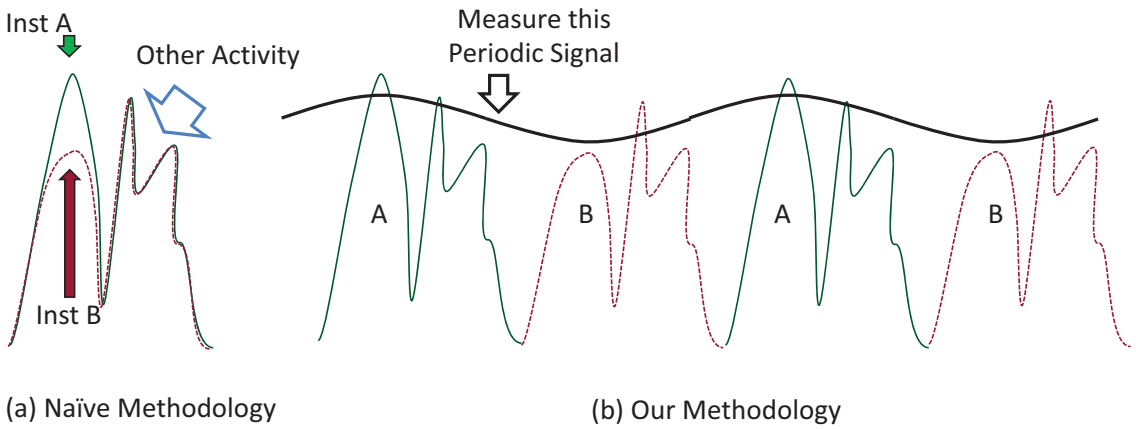

Figure 3. Our methodology measures the (a) signal difference by (b) alternating the signals then filtering and measuring the resulting periodic signal at the alternation frequency.

be accumulated to mount a successful attack depends on the SAVAT values between these instructions – huge SAVAT values enable attacks even when sensitive data creates a seemingly small difference in execution, e.g. the attacker may need fewer such "loud" instructions.

Single instruction differences in execution may be accumulated in two ways: 1) repetition: the same single-instruction difference may be re-created many times, and the attacker can use the overall difference that is created, and 2) combination: entire sequences of different instructions can be executed. Our measurement methodology exploits repetition to obtain signals that can be more reliably measured, then divides the large measured signal by the number of repetitions to determine the contribution of a single instance. Combination is not directly addressed in this paper – while we believe that the sum of single-instruction differences can act as a good estimate for the combined signal, this estimate is imprecise because instructions can be reordered and their execution may overlap. A more accurate SAVAT measurement of signal differences created by executing different sequences of instructions can be performed by using those entire sequences as A/B activity in the measurement. However, this approach does not scale well to longer sequences: pairwise SAVAT measurement for N individual instructions requires $O(N^2)$ measurements, pairwise measurement among all possible two-instruction sequences constructed from these N instructions requires $O(N^4)$ measurements, etc. Ways of addressing this combinatorial explosion problem are a priority for our future research. One approach is to cluster instruction opcodes using SAVAT as the distance metric, then explore sequences using instruction class representatives. Another approach would be to derive a good model of the interaction among instructions in a sequence, i.e. to capture effects of reordering, dependences, etc., and then compute overall SAVAT values for instruction sequences by using the interaction model to combine measured single-instruction SAVAT values. Such models are beyond the scope of this paper, but they are an important aspect for our future work.

```
1  while(1){
2    // Do some instances of the A inst/event
3    for(i=0;i<inst_loop_count;i++){
4      ptr1=(ptr1&~mask1)|((ptr1+offset)&mask1);
5      // The A-instruction, e.g. a load
6      value=*ptr1;
7    }
8    // Do some instances of the B inst/event
9    for(i=0;i<inst_loop_count;i++){
10     ptr2=(ptr2&~mask2)|((ptr2+offset)&mask2);
11     // The B-instruction, e.g. a store
12     *ptr2=value;
13   }
14 }
```

Figure 4. The A/B alternation pseudo-code.

The overall structure of the code used in our new measurement methodology is shown in Figure 4. Lines 2 through 7 execute `inst_loop_count` instances of the A instruction/event, and then lines 8 through 13 execute the same number of instances of the B instruction. Thus lines 2 through 13 represent one A/B alternation, and this alternation is repeated (line 1) until the measurement of the side channel signal is complete. The value of `inst_loop_count` allows us to control the number of alternations per second, and we select a value that produces the desired alternation frequency for our measurements.

To generate different cache behavior in load and store instructions, our code (in lines 4 and 10) updates the address of the accessed location so the memory access repeatedly sweeps over an array of appropriate size (fits in L1 cache, does not fit in L1 but fits in L2 cache, or does not fit in L2) to create the desired cache hit/miss behavior. Note that `ptr1`, `ptr2`, and `offset` must be chosen so that the A and B instructions access separate groups of cache blocks to create the desired cache behavior (e.g. every A instruction is a L1 cache hit and every B instruction is a

L2 cache hit). Aside from the test instructions (line 6 for A and line 12 for B), the executed code should be identical for all instructions/events, so this pointer-update code is present even when the A and/or B instruction is a non-memory instruction (e.g. ADD). Our actual code is written in x86 assembler to minimize the amount of non-under-test activity and prevent compiler optimizations that might make the non-under-test code differ for different under-test instructions (e.g. different instruction scheduling by the compiler, dead code elimination of memory address updates for non-memory instructions, etc.).

In developing this measurement methodology, we made two important simplifying assumptions about execution of the test code in a modern system. First, we assume that the observed signal is due to the differences between test instructions A and B (lines 6 and 12) and not due to differences in how the processor executes surrounding code (which contains identical instructions for both A and B activity). In Section V we present results that support this assumption to a large degree - the signals created by alternation of the same activity (A/A instead of A/B) are very weak compared to the largest A/B SAVAT values we observe. The second assumption is that the signal created by the alternation will be in a narrow frequency band around the alternation frequency, which allows this signal to "stick out" above the measurement and environment noise. Our measurements confirm that this assumption does hold – one of the most frequency-dispersed signals we observe is shown in Figure 7, and it is still contained in a narrow frequency range.

IV. Experimental Setup for EM Side Channel Measurements

	Instruction	Description
LDM	mov eax,[esi]	Load from main memory
STM	mov [esi],0xFFFFFFFF	Store to main memory
LDL2	mov eax,[esi]	Load from L2 cache
STL2	mov [esi],0xFFFFFFFF	Store to L2 cache
LDL1	mov eax,[esi]	Load from L1 cache
STL1	mov [esi],0xFFFFFFFF	Store to L1 cache
ADD	add eax,173	Add imm to reg
SUB	sub eax,173	Sub imm from reg
MUL	imul eax,173	Integer multiplication
DIV	idiv eax	Integer division
NOI		No instruction

Figure 5. x86 instructions for our A/B SAVAT measurements.

In this case study we constructed the A/B alternation code as described in Section III for each pairwise combination of the eleven instructions listed in Figure 5. These include loads and stores serviced by different levels of the cache hierarchy, simple (ADD and SUB) and more complex (MUL and DIV) integer arithmetic, and the "No instruction" case where the appropriate line in our alternation code (Line 6 or 12 in Figure 4) is simply left empty.

Processor	L1 Data Cache	L2 Cache
Intel Core 2 Duo	32 KB, 8 way	4096 KB, 16 way
Intel Pentium 3 M	16 KB, 4 way	512 KB, 8 way
AMD Turion X2	64 KB, 2 way	1024 KB, 16 way

Figure 6. Laptop systems measured in our case study.

This code is run as a single-threaded 32-bit user mode console application on three different laptop systems shown in Figure 6. At the time of testing, no other applications were active and wireless devices were disabled to minimize interference with measured signals. Aside from this, the system was operating normally, meaning that any EM signals resulting from system processes and other OS activity would affect the received signal. The resulting periodic EM signal was measured using a magnetic loop antenna (AOR LA400) connected to a spectrum analyzer (Agilent MXA N9020A). A recent paper illustrates a practical EM attack on implementations of the RSA and ElGamal algorithms using similar measurement instruments and computer systems [22]. In our study several measurements were taken to observe the variation between systems and reception distances. All three systems were measured with an A/B alternation frequency of 80 kHz and a measurement distance of 10 cm. Additional measurements were performed on the Core 2 Duo laptop with the 80 kHz alternation frequency at laptop-to-antenna distances of 50 cm and 100 cm.

The spectrum around the alternation frequency was recorded with a resolution bandwidth of 1Hz, which results in a very low measurement noise floor because the measured signal is affected only by noise from a 1Hz-wide spectral band. The actual recorded spectrum for an ADD/LDM (add instruction and off-chip load instruction) measurement is shown in Figure 7.

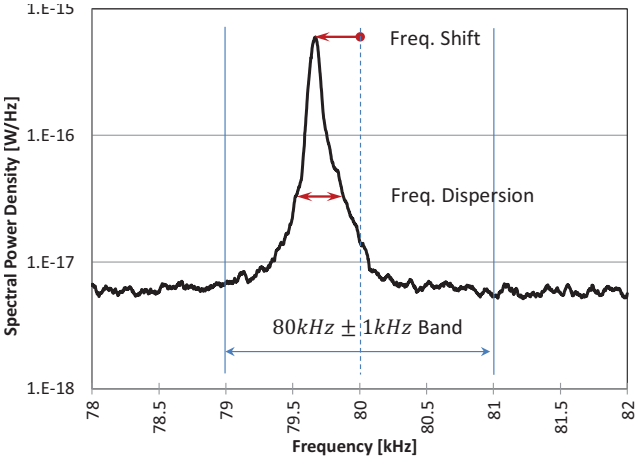

Figure 7. Recorded spectrum for 80 kHz ADD/LDM alternation.

Because the alternation frequency in the running code cannot be controlled perfectly, the actual measured signal is not perfectly concentrated at the intended alternation frequency, which was 80 kHz for Figure 7. The actual alternation

frequency can be slightly different – in Figure 7 it is shifted lower by about 400 Hz, and it can vary during the measurement so it is dispersed around the central frequency (as indicated in Figure 7). Thus the measured value we use is the total received signal power in the frequency band from 1 kHz below to 1 kHz above the alternation frequency (illustrated near the bottom of Figure 7). In each A/B measurement, the same number of A and B instructions is executed per second, so we divide this total signal power by the number of executed A/B pairs. Because power (energy/second) is divided by the instruction-pairs/second, the resulting SAVAT quantity is actually the signal energy available to the attacker to discern whether a single A or a single B instruction was executed. Finally, each measurement campaign to gather the 11-by-11 pairwise A/B SAVAT was repeated 10 times over a period of multiple days to assess how the measurement is affected by changes in radio signal interference, room temperature, errors in positioning the antenna, etc.

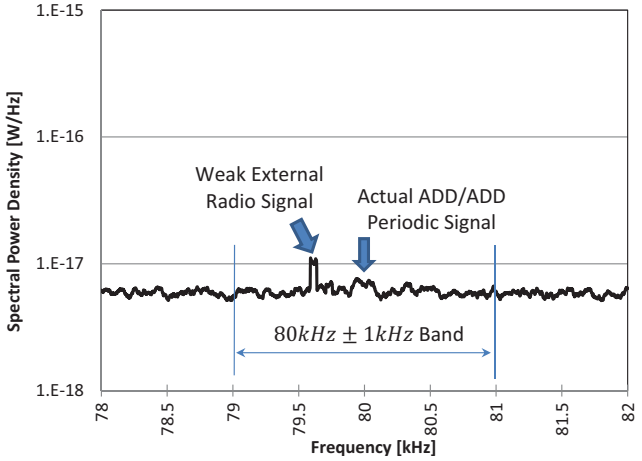

Figure 8. Recorded spectrum for 80 kHz ADD/ADD alternation.

Our measurements include all 11 cases where A and B are the exact same instruction/event, where the resulting A/A alternation should result in no signal at the alternation frequency. An actual measured spectrum for ADD/ADD is shown in Figure 8. We see that some signal does exist in the band around the intended alternation frequency: the instrument's sensitivity floor (which is around 6×10^{-18} W/Hz in Figure 8), external radio signals, and a weak signal created by imperfect matching of A/B not-under-test activity. Therefore, these same-instruction alternation measurements give us a very good estimate of the experimental measurement error, and can help identify possible problems such as strong radio interference or mistakes in the A/B alternation code. When A and B instructions/events are not the same, we measure both the A/B alternation and the B/A alternation - these should be the same, so their difference allows us to assess the measurement error caused by placing identical instructions at different program addresses, i.e. the effect of fetch-related variations such as instruction cache alignment.

V. Experimental Results for EM Side Channel Measurements

A. *Measurements at* 10 cm *Distance*

Each measurement campaign results in a 11-by-11 matrix of pairwise A/B SAVAT values for a particular system, alternation frequency, and antenna distance. The matrix for the Core 2 Duo laptop with the 10 cm distance and 80 kHz intended alternation frequency is shown in Figure 9. Note that these values are extremely small - they are in zepto-joules (1zJ = 10^{-21}J)! This indicates that one occurrence of a single-instruction difference would probably not be sufficient for the attacker to decide which of the two instructions was executed – many repetitions of the same instruction, or many instructions worth of difference will be needed. Unfortunately, repetition is common for some kinds of sensitive data, e.g. a cryptographic key can be reused many times while encrypting a long stream of data.

	LDM	STM	LDL2	STL2	LDL1	STL1	NOI	ADD	SUB	MUL	DIV
LDM	1.8	2.4	7.9	11.5	4.6	4.4	4.3	4.2	4.4	4.2	5.1
STM	2.3	2.4	8.8	11.8	4.3	4.2	3.8	3.9	3.9	4.3	4.2
LDL2	7.7	7.7	0.6	0.8	3.9	3.5	4.3	3.6	4.8	3.8	6.2
STL2	11.5	10.6	0.8	0.7	5.1	6.1	6.1	6.1	6.1	6.2	10.1
LDL1	4.4	4.2	3.3	5.8	0.7	0.6	0.7	0.7	0.7	0.7	1.3
STL1	4.5	4.2	3.8	4.9	0.7	0.6	0.7	0.6	0.6	0.6	1.2
NOI	4.1	3.8	4.1	6.4	0.7	0.7	0.6	0.6	0.7	0.6	1.0
ADD	4.2	4.1	4.1	7.0	0.7	0.7	0.6	0.7	0.6	0.6	1.0
SUB	4.4	4.0	3.8	7.3	0.7	0.6	0.7	0.6	0.6	0.6	1.1
MUL	4.4	3.9	3.7	5.7	0.7	0.7	0.6	0.6	0.6	0.6	1.1
DIV	5.0	4.6	6.9	9.3	1.3	1.2	1.0	1.1	1.1	1.1	0.8

Figure 9. SAVAT values (in zJ) for the Core 2 Duo laptop at the 10 cm distance and at the 80 kHz intended alternation frequency.

Figure 9 shows large variations in SAVAT among these instruction pairs – this means that some instruction pairs are much easier for attackers to disambiguate than others. Each entry in this table is the mean for a set of 10 measurements where the A instruction is given by the row and the B instruction is given by the column. This table can also be used to confirm some assumptions about our measurement methodology. First, each of the diagonal entries in the table (the A/A pairs) is the smallest value in its respective row and column (with one exception for STM/LDM). This validates the assumption that the largest (i.e. most interesting/dangerous) measured SAVAT values are predominantly a result of actual differences among instructions under consideration, and not of the surrounding code that should be the same for all instructions under test.

Each SAVAT value in Figure 9 is an average of ten separate measurements. To investigate how repeatable our measurements are, we compute the standard deviation for each such set of ten measurements. We found that the standard-deviation-to-mean ratio is 0.05 on average, i.e. individual measurements vary about 5% around the ten-measurement mean. This implies that our measurements are repeatable, and indicates that the signal created by the

alternation loop (discussed in the previous paragraph) is the dominant source of error in the measured SAVAT values.

Figure 10 visualizes the mean values of Figure 9 as shades of gray, where white corresponds to the smallest and black to the largest SAVAT values, and Figure 11 shows SAVAT values for selected instruction pairings as a traditional bar chart.

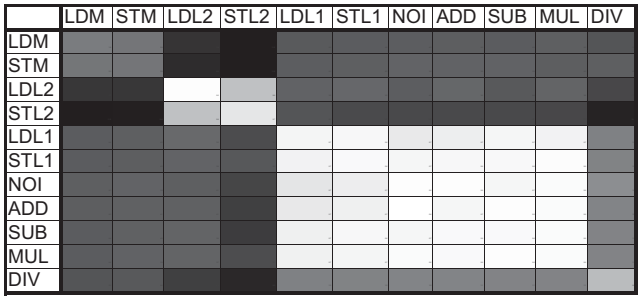

Figure 10. Visualization of SAVAT values from Figure 9.

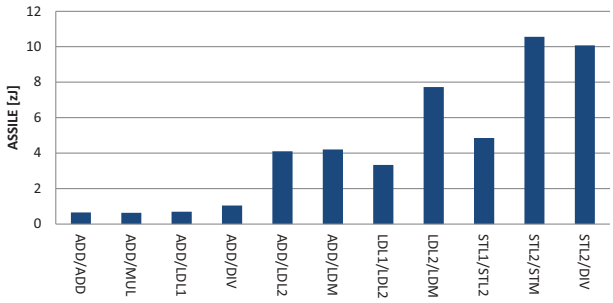

Figure 11. SAVAT for selected instruction pairings from Figure 9.

We observe four groups of instructions/events that have low intra-group and high inter-group SAVATs: *The off-chip access group* (LDM and STM), *the L2 hit group* (LDL2 and STL2), *an Arithmetic/L1 group that includes ADD, SUB, MUL, NOI, and also LDL1 and STL1*, and *a group that only contains the DIV instruction*. We can see that the SAVAT between instructions in the Arithmetic/L1 group is similar to the same-instruction measurement (e.g. ADD/ADD), i.e. it is very difficult for attackers to distinguish between instructions in this group. Although their functionality is quite different, L1 cache accesses are also very difficult to distinguish from ADD/SUB/MUL arithmetic instructions. As expected, L2 accesses and main-memory accesses are much easier to distinguish from other instructions. Note that an L2 store hit is noticeably easier to distinguish from other instructions than it is an L2 load hit. This might be caused by the fact that we cannot create a sustained string of L1 write misses without also creating dirty replacements from L1 to L2, i.e. each STL2 instruction creates two L2 accesses - one to fetch the block from the L2 cache into L1, and later another that writes back the dirty block from L1 to L2. So the higher SAVAT values for STL2 might be attributable to write-back activity caused by these instructions. If that is true, however, it is surprising that STM does not have higher SAVAT values than LDM, even though it includes write-back activity that LDM does not have.

Also surprisingly, the DIV instruction has noticeably higher SAVAT values than ADD, SUB, and MUL. It is also surprising that off-chip memory accesses and L2 hits have similar SAVAT, i.e. the task of distinguishing between LDM and ADD using EM emanations is similar in difficulty to the task of distinguishing between LDL2 and ADD. This is contrary to the intuitive expectation that off-chip accesses should create stronger emanations because they toggle long off-chip wires that can act as better transmission antennae for EM emanations. Interestingly, however, off-chip memory accesses do have an even higher SAVAT when paired with L2 hits than when paired with other instructions. One possible explanation for this is that e.g. LDM creates an EM fields that allows it to be distinguished from e.g. an ADD, and that LDL2 creates an EM field that is similarly distinguishable from an ADD, but the fields for LDM and LDL2 are different from each other and very easy to distinguish.

For computer architects who desire to reduce the potential for EM side channel attacks on their processors, these results indicate that the path of least resistance for the attackers is in code that uses off-chip accesses, L2 cache accesses, and possibly DIV instructions in ways that depend on sensitive data, so the architects' focus should be on making execution of these instructions less EM-noisy, e.g. through limited use of compensating-activity techniques. For programmers, these results confirm what programmers should already know from work on other side channels - in code that processes sensitive data, special care should be taken to avoid situations where a memory access instruction might have an L2 hit or miss depending on the value of some sensitive data item. Code that does not have data-dependent variation in cache hit/miss behavior is considerably less vulnerable to EM side channel attacks, and the most worrisome situation in that code would be one where a DIV instruction is executed or not depending on sensitive data, e.g. when a control flow decision based on sensitive data selects between a path that includes a DIV instruction and another that does not.

Figures 12 and 13 show the results of a similar measurement (10 cm, 80 kHz) for the Pentium 3 M laptop, which is several processor generations older than the Core 2 Duo laptop. Some of the trends in these figures are similar - the ADD/SUB/MUL instructions are very difficult to distinguish from each other, the SAVAT for pairings of L2 accesses and arithmetic instructions is higher (and similar to what we saw for the Core 2 Duo laptop), and the DIV instruction has higher SAVAT than other arithmetic instructions. However, in this laptop the DIV instruction is *much* easier to distinguish from other arithmetic instructions - the ADD/DIV SAVAT is an order of magnitude higher

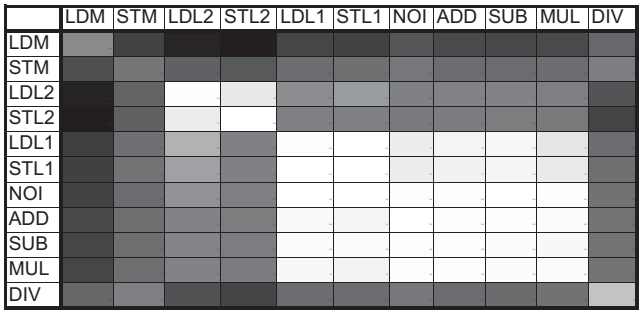

Figure 12. Visualization of SAVAT for the Pentium 3 M laptop.

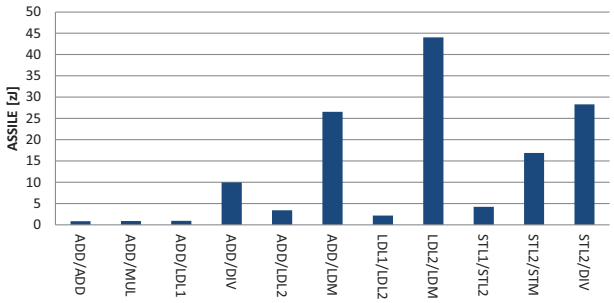

Figure 13. SAVAT for selected instruction pairings for the Pentium 3 M laptop.

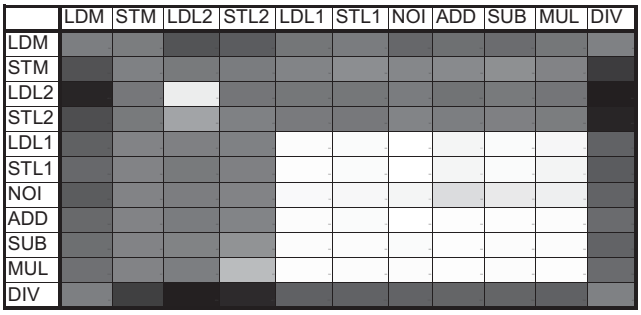

Figure 14. Visualization of SAVAT for the Turion X2 laptop.

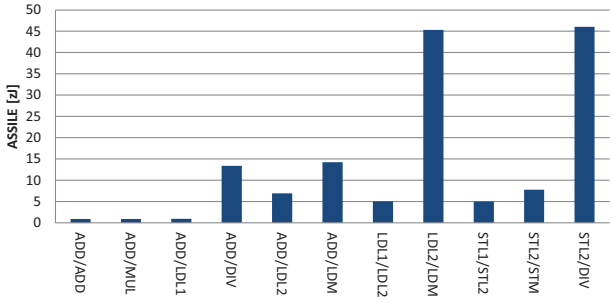

Figure 15. SAVAT for selected instruction pairings for the Turion X2 laptop.

than the ADD/MUL SAVAT. Similarly, off-chip accesses here have much higher SAVAT values than do L2 accesses, and LDM has higher SAVAT values than STM. Overall, it seems that the high-SAVAT problem of DIV and off-chip load/store instructions in the Pentium 3M processor was reduced when designing Core 2 (released 7 years after the Pentium 3 M). It is possible that the reason for this improvement was not a deliberate effort to alleviate EM side channel vulnerabilities – reduction in EM leakage might be a side effect of a reduction in operating voltages, shorter wire lengths in the technology-shrunk divider, and signaling optimizations that save power by reducing wire toggling at the processor-memory interface.

Finally, Figures 14 and 15 show the results of a similar measurement (10 cm, 80 kHz) for the Turion X2 laptop, whose processor was released in the same year as Core 2 Duo. We observe very similar results here as we did for the Pentium 3 M, except that the DIV instruction here has an even higher SAVAT values - they rival those of off-chip memory accesses.

Overall, we observe both similarities and differences among different systems, so SAVAT results can be useful to both 1) architects who wish to identify (and alleviate) the aspects of their processor and system that are the most susceptible to EM side channel vulnerabilities, especially those that are endemic to that processor, and 2) software developers who need to know which variations in program behavior are the most likely to allow successful side channel attacks, especially for behaviors that are consistently vulnerable across several generations of processors and among several processor manufacturers.

B. *Measurements at* 50 cm *and* 100 cm *Distances*

The EM side channel has an interesting parameter that other channels do not - the distance from which the attack is attempted. Thus we performed additional measurements using the Core 2 Duo laptop, with the antenna at distances of 50 cm and 100 cm from the laptop.

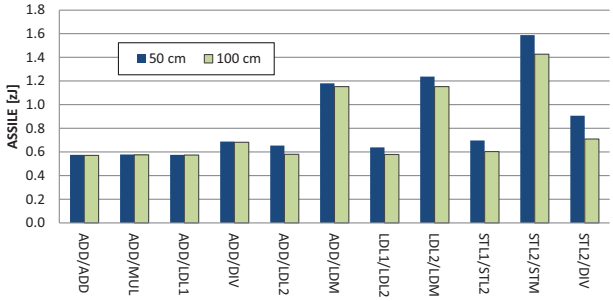

Figure 16. SAVAT measured at 50 cm and 100 cm for the Core 2 Duo laptop.

Figure 16 shows the measured SAVAT values for selected instruction pairings at these distances. As expected, we observe significantly lower SAVAT values for these larger distances but, interestingly, the SAVAT values do not drop much when changing the distance from 50 cm to 100 cm. Another interesting aspect of these measurements is that at these larger distances pairings that include off-chip activity

have higher SAVAT values than pairings of on-chip events; although pairings of DIV with other instructions do have slightly higher SAVAT than do pairings that include ADD, SUB, or MUL, this difference is now very small.

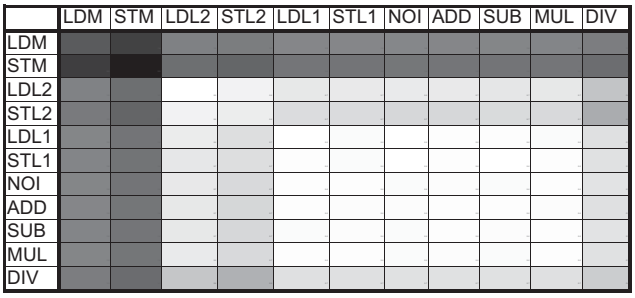

Figure 17. Visualization of SAVAT results at 50 cm.

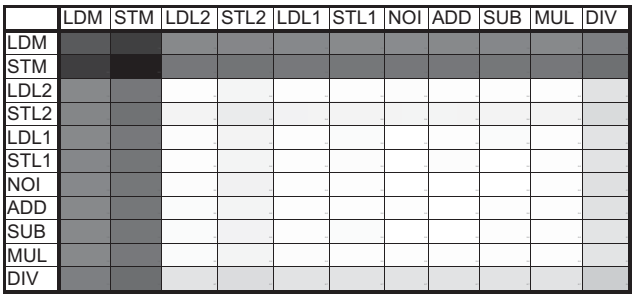

Figure 18. Visualization of SAVAT results at 100 cm.

Figures 17 and 18 show a visualization of SAVAT values measured from distances of 50 cm and 100 cm, respectively. We observe that off-chip memory accesses are now (by far) the most attacker-distinguishable type of instruction/event - although the SAVAT of both LDM and STM when alternated with other instructions is significantly lower than at 10 cm distance, the SAVAT values for LDL2 and STL2 have dropped a lot more with this distance change. This indicates that vulnerability to EM side channel attacks must be assessed at distances similar to those that might be used by attackers. If only long-range (e.g. from a neighboring desk in an office) attacks are realistic but we use a very short measurement distance, we might be alleviating EM signals that attenuate quickly with distance, such as near-field signals, that are quickly attenuated with distance and are not such a significant threat for long-range attacks. On the other hand, if short-distance attacks are possible, e.g. if the attacker attaches the receiver to the underside of our desk surface, use of only long-range measurements might lead us to ignore some potential short-distance vulnerabilities.

We note that the need to select an appropriate measurement distance, or to measure at several different distances, does not detract from the usefulness of our SAVAT metric or from our repeated-alternation measurement methodology: any metric and any measurement approach can be misused by measuring under one set of conditions and applying the results under very different conditions.

VI. Related Work

Side channel analysis has emerged as a powerful tool for data theft, especially in the context of breaking commercial cryptography implementations. Such analysis exploits the fact that cryptography implementations on physical devices transfer much more information than just the desired input-output information [33], [34]. In cryptography, a side channel attack is defined as any attack that uses information gained or leaked from the physical implementation of a system, and this definition has been used for other data theft contexts as well. Side channel attacks can be classified as timing attacks, power monitoring attacks, electromagnetic attacks, acoustic attacks, differential fault attacks, bus-snooping attacks, cache-based attacks, etc. Timing and power attacks are based on measuring the time and power that system needs to perform particular operations (e.g., encryption operations) [6], [10], [14], [25], [33], [34], [43]. Electromagnetic attacks are based on leaked electromagnetic radiation [2], [3], [32], and acoustic attacks exploit the sound (acoustic "noise") produced by a system as it carries out computation [4], [12], [51]. Differential fault attacks are based on using the secrets discovered by introducing faults in a computation [9], [24]. Cache-based attacks extract secrets by observing (using a malicious process running on the same machine) which memory blocks are accessed by the target application [5], [61], [63]. Other microarchitecture-based side channel attacks have been reported, e.g. attacks based on branch prediction or instruction caches [1].

Different side channels offer different advantages and disadvantages to potential attackers. Timing and power side channel attacks [6], [10], [11], [18], [25], [33], [34], [43], [49] only require relatively simple equipment. Power attacks are relatively easy to detect because measurement equipment must be attached to the target system, while timing attacks require attackers to interact (e.g. request-response over the network) with the system or observe other users' interactions. Bus snooping attacks require physical access to the system, but they are powerful because they give the attacker direct access to observe (and even modify) values that are communicated on the processor-memory bus [30], [29] or that are stored in the system's memory [36]. Bus snooping attacks are well understood and effective targeted solutions have been proposed in research literature [13], [21], [23], [37], [38], [39], [40], [42], [48], [52], [53], [54], [55], [56], [57], [58], [64], [65], [68] and used in actual products [31], [41]. Cache-based attacks and other resource-sharing-based attacks do not require physical access, but they do require running the attacker's code on the same system that runs the "victim" code. These attacks can be defended against by both traditional measures (prevent malicious code from being introduced) and by careful hardware design [63].

Electromagnetic side channel attacks are very difficult to detect because they can be mounted from a distance. The existence of side channel EM emanations was reported in the open literature as early as 1966 [28], [66], and even older classified TEMPEST work is often discussed [32], [66]. Analysis of security risks from EM emanations from computer monitors has been presented in 1985 [62], and low-cost software techniques for reducing risks from computer display emanations have been proposed much later [32] and mostly apply to now-obsolete CRT displays. Radio transmission through EM emanations from a desktop's memory system was detected at a distance of about two meters [17] as a proof-of-concept experiment, but without any analysis or direction toward potential solutions. Some systematic investigation of leakage of smartcard EM emanations [2], [20] classifies EM emanations as direct emanations, caused by current flows within circuits (e.g. switching activity while adding two numbers in a processor), and indirect emanations due to EM couplings among chip circuits (e.g. activity on a data bus coupling to and modulating the amplitude of a nearby clock signal). Recently reported results for modern systems [67] show that information can be transmitted through EM emanations to distances of at least 2-3m, even in the presence of significant countermeasures (metal shielding, walls, etc.), and practical EM attacks against implementations of the RSA and ElGamal algorithms running on laptops have been reported [22]. Attacks against cryptographic hardware [26] and keyboards [35] have also been reported. Countermeasures for EM leakage have been proposed for smartcards [19], [45], [46], [47], [59], [60], including the use of asynchronous circuits [19], low-cost shielding (e.g. metal foil) [45], transmission of jamming signals [46], etc. Although these proposed solutions do attenuate EM emanations from a system, they are presumably more expensive in terms of silicon cost, weight and bulk, and/or power consumption compared to solutions that directly address the leakage mechanisms.

Solutions for other problems may have an impact (positive or negative) on side channel signals. For example, significant research and applied work on EM interference and compatibility (EMI/EMC) [27], [44] also aims to reduce the spectral density of EM emanations. Some methods, such as shielding, alleviate both EMC and EM side channel issues. Some methods, however, help one at the expense of other. For example, transmission of jamming signals [46] may help mask EM side channel signals but negatively affects EMC. Recent findings show that EM signals from computer systems may satisfy EMC regulations but still be detected in EM side channel attacks [50].

The Side-Channel Vulnerability Factor (SVF) [15], [16] measures the correlation between high-level execution patterns (e.g. program phase transitions) and side channel signals. This provides high level information about potential leakage from programs, but does not provide the instruction level information that would be needed to attribute the potential leakage to specific processor/system components or to specific sites (e.g. line of code) in a program. Previous research has developed methods for measuring energy per instruction (for example [7]), however more work is needed to determine how differences in the expenditure of energy propagate to the attacker to use these measurements in the context of side channel analysis. Whereas previous work measures the energy *expended* per instruction, the metric discussed in this paper measures only the energy that can be *received* and *exploited* by an attacker through a given side channel (EM emanations in our case study).

Overall, our instruction-level metric and methodology differ from prior work in that we *quantify* the signal that is sent to the attacker by an *instruction-level difference* in program execution. These measurements can be used to determine the potential for information leakage when execution of individual instructions or even sections of code depends on sensitive information. We expect our instruction-level attribution of potential side channel vulnerability to help system designers decide *where in the system/processor* to apply countermeasures, and also to help programmers and compilers apply software-based countermeasures selectively to minimize their performance and power impact.

VII. Conclusion

Side channels are a powerful class of attacks that circumvent traditional protections, and a significant number of such attacks and potential countermeasures have been proposed for both hardware and software. Recent work has shown that potential side channel vulnerability can be assessed at the level of an entire processor or system, and at the granularity of entire phases. However, without a practical way to attribute potential side channel vulnerability to specific instruction-level behavior, it is difficult for computer architects and software developers to apply countermeasures strategically to limit their cost and performance/power impact.

This paper presents a new metric, which we call Signal Available to Attacker (SAVAT), that measures the side channel signal created by a specific single-instruction difference in program execution, i.e. the amount of signal made available to a potential attacker who wishes to decide whether the program has executed instruction/event A or instruction/event B. We also devise a practical methodology for measuring SAVAT in real systems using only user-level access permissions and realistic measurement equipment, and perform a case study where we measure EM emanations SAVAT among 11 different instructions for three different laptop systems. Our findings from these experiments confirm key intuitive expectations, e.g. that SAVAT between on-chip instructions and off-chip memory accesses tends to be higher than between two on-chip instructions, but we also find that at short distances particular instructions, such as the integer

divide, have much higher SAVAT than other instructions in the same general category (integer arithmetic), and that last-level-cache hits and misses have similar (high) SAVAT. Overall, we confirm that our new metric and methodology can help discover the highest-vulnerability aspects of a processor architecture or a program, and thus inform decision-making about how to best manage the overall side channel vulnerability of a processor, program, or system.

Although we have demonstrated that our instruction-level metric and methodology do lead to useful insights, we also believe that this is just a first step toward potentially more informative fine-grain metrics, and more refined measurement methodologies. Another direction for future research is to measure SAVAT for multiple side channels to help inform decisions about which ones are the most dangerous for a particular class of processors or systems. Strategies for systematically classifying and clustering instructions and events without the need to measure SAVAT for all pairwise combinations would also be helpful, and probably necessary to "cover" large instructions sets such as x86. Finally, our methodology can be extended to assess the side channel signals generated by microarchitectural activity beyond data cache hits/misses. Examples that may have high SAVAT and should be studied include branch prediction hit/misses, bursts of wakeup/select activity, and coherence activity in multi-core systems.

VIII. Acknowledgments

This work has been supported, in part, by NSF grant 1318934 and AFOSR grant FA9550-14-1-0223. The views and findings in this paper are those of the authors and do not necessarily reflect the views of NSF or AFOSR.

References

[1] O. Aciiçmez, c. K. Koç, and J.-P. Seifert, "On the power of simple branch prediction analysis," in *Proc. 2nd ACM Symp. on Information, Computer and Communications Security*. ACM Press, Mar. 2007, pp. 312–320.

[2] D. Agrawal, B. Archambeult *et al.*, "The EM side-channel(s)," in *Proc. Crypto. HW and Emb. Sys. (CHES)*, 2002, pp. 29–45.

[3] ——, "The EM side-channel(s): attacks and assessment methodologies," 2002. [Online]. Available: http://www.research.ibm.com/intsec/emf-paper.ps

[4] M. Backes, M. Durmuth *et al.*, "Acoustic side-channel attacks on printers," in *Proc. USENIX Security Symp.*, 2010.

[5] E. Bangerter, D. Gullasch, and S. Krenn, "Cache games - bringing access-based cache attacks on AES to practice," in *Proc. IEEE Symp. on Security and Privacy*, 2011.

[6] A. G. Bayrak, F. Regazzoni *et al.*, "A first step towards automatic application of power analysis countermeasures," in *Proc. 48th Design Automation Conf.*, 2011.

[7] R. Bertran, A. Buyuktosunoglu *et al.*, "Systematic energy characterization of cmp/smt processor systems via automated micro-benchmarks," in *Proc. 45th Int'l Symp. on Microarchitecture*, 2012, pp. 199–211.

[8] E. Biham and A. Shamir, "Differential cryptanalysis of the data encryption standard," in *Proc. Int'l Cryptology Conf.*, 1993.

[9] ——, "Differential fault analysis of secret key cryptosystems," in *Proc. Int'l Cryptology Conf.*, 1997, pp. 513–525.

[10] D. Boneh and D. Brumley, "Remote Timing Attacks are Practical," in *Proc. USENIX Security Symp.*, 2003.

[11] S. Chari, C. S. Jutla *et al.*, "Towards sound countermeasures to counteract power-analysis attacks," in *Proc. Int'l Cryptology Conf.*, 1999, pp. 398–412.

[12] S. Chari, J. R. Rao, and P. Rohatgi, "Template attacks," in *Proc. Crypto. HW and Emb. Sys. (CHES)*, 2002, pp. 13–28.

[13] D. Clarke, S. Devadas *et al.*, "Checking the Integrity of Memory in a Snooping-Based Symmetric Multiprocessors," *MIT-CSAIL Memo No. 470*, 2004.

[14] B. Coppens, I. Verbauwhede *et al.*, "Practical Mitigations for Timing-Based Side-Channel Attacks on Modern x86 Processors," in *Proc. IEEE Symp. on Security and Privacy*, 2009, pp. 45–60.

[15] J. Demme, R. Martin *et al.*, "Side-channel vulnerability factor: A metric for measuring information leakage," in *Proc. 39th Int'l Symp. on Comp. Arch. (ISCA)*, 2012, pp. 106–117.

[16] ——, "A quantitative, experimental approach to measuring processor side-channel security," *Micro, IEEE*, vol. 33, no. 3, pp. 68–77, May 2013.

[17] B. Durak, "Controlled CPU TEMPEST Emanations," 1999. [Online]. Available: http://cryptome.org/tempest-cpu.htm

[18] P. N. Fahn and P. J. Pearson, "A new class of power attacks," in *Proc. Crypto. HW and Emb. Sys. (CHES)*, 1999, pp. 173–186.

[19] J. J. A. Fournier, S. Moore *et al.*, "Security evaluation of asynchronous circuits," in *Proc. Crypto. HW and Emb. Sys. (CHES)*, 2003, pp. 137–151.

[20] K. Gandolfi, C. Mourtel, and F. Olivier, "Electromagnetic analysis: concrete results," in *Proc. Crypto. HW and Emb. Sys. (CHES)*, 2001, pp. 251–261.

[21] B. Gassend, G. Suh *et al.*, "Caches and Hash Trees for Efficient Memory Integrity Verification," in *Proc. 9th Int'l Symp. on High-Perf. Comp. Arch. (HPCA)*, 2003.

[22] D. Genkin, I. Pipman, and E. Tromer, "Get Your Hands Off My Laptop: Physical Side-Channel Key-Extraction Attacks on PCs," in *Proc. Crypto. HW and Emb. Sys. (CHES)*, 2014.

[23] T. Gilmont, J.-D. Legat, and J.-J. Quisquater, "Enhancing the Security in the Memory Management Unit," in *Proc. 25th EuroMicro Conference*, 1999.

[24] C. Giraud, "DFA on AES," in *Proc. 4th Int'l AES Conf.* Springer, 2003, pp. 27–41.

[25] L. Goubin and J. Patarin, "DES and Differential power analysis (the duplication method)," in *Proc. Crypto. HW and Emb. Sys. (CHES)*, 1999, pp. 158–172.

[26] Y.-I. Hayashi, N. Homma *et al.*, "Transient analysis of em radiation associated with information leakage from cryptographic ics," in *9th Int'l Workshop on EM Compat. of ICs (EMC Compo)*, 2013, pp. 78–82.

[27] Henry W. Ott, *Electromagnetic Compatibility Engineering*. Wiley, 2009.

[28] H. J. Highland, "Electromagnetic radiation revisited," *Computers and Security*, pp. 85–93, Dec. 1986.

[29] A. B. Huang, "The Trusted PC: Skin-Deep Security," *IEEE Computer*, vol. 35, no. 10, pp. 103–105, 2002.

[30] ——, *Hacking the Xbox: An Introduction to Reverse Engineering*. No Starch Press, San Francisco, CA, 2003.

[31] IBM, "IBM Extends Enhanced Data Se-

curity to Consumer Electronics Products," *http://domino.research.ibm.com/comm/pr.nsf/pages/news.20060410_security.html*.

[32] M. G. Khun, "Compromising emanations: eavesdropping risks of computer displays," *The complete unofficial TEMPEST web page: http://www.eskimo.com/˜joelm/tempest.html*, 2003.

[33] P. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems," in *Proc. Int'l Cryptology Conf.*, 1996, pp. 104–113.

[34] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis: leaking secrets," in *Proc. Int'l Cryptology Conf.*, 1999, pp. 388–397.

[35] M. Kuhn, "Compromising emanations of lcd tv sets," *IEEE Trans. Electromagn. Compat.*, vol. 55, no. 3, pp. 564–570, June 2013.

[36] A. Kumar, "Discovering Passwords in Memory," *http://www.infosec-writers.com/text_resources/*, 2004.

[37] M. Lee, M. Ahn, and E. J. Kim, "I2SEMS: Interconnects-Independent Security Enhanced Shared Memory Multiprocessor Systems," in *Proc. Int'l Conf. on Paral. Arch. and Comp. Techn. (PACT)*, 2007.

[38] R. B. Lee, P. C. S. Kwan *et al.*, "Architecture for Protecting Critical Secrets in Microprocessors," in *Proc. 32nd Int'l Symp. on Comp. Arch. (ISCA)*, 2005.

[39] D. Lie, J. Mitchell *et al.*, "Specifying and Verifying Hardware for Tamper-Resistant Software," in *Proc. IEEE Symp. on Security and Privacy*, 2003.

[40] D. Lie, C. A. Thekkath *et al.*, "Architectural Support for Copy and Tamper Resistant Software," in *Proc. 9th Int'l Conf. on Arch. Support for Prog. Lang. and Op. Sys. (ASPLOS)*, 2000.

[41] Maxim/Dallas Semiconductor, "DS5002FP Secure Microprocessor Chip," *http://www.maxim-ic.com/quick_view2.cfm/qv_pk/2949*.

[42] J. P. McGregor and R. B. Lee, "Protecting Cryptographic Keys and Computations via Virtual Secure Coprocessing," in *Workshop on Arch. Sup. for Sec. and Anti-virus (WASSA)*, 2004.

[43] T. S. Messerges, E. A. Dabbish, and R. H. Sloan, "Power analysis attacks of modular exponentiation in smart cards," in *Proc. Crypto. HW and Emb. Sys. (CHES)*, 1999, pp. 144–157.

[44] C. R. Paul, *Introduction to Electromagnetic Compatibility*, 2nd ed. Wiley, 2006.

[45] T. Plos, M. Hutter, and C. Herbst, "Enhancing side-channel analysis with low-cost shielding techniques," in *Proceedings of Austrochip*, 2008.

[46] F. Poucheret, L. Barthe *et al.*, in *Proc. 18th VLSI System on Chip Conf.*

[47] J. J. Quisquater and D. Samyde, "Electromagnetic analysis (EMA): measures and counter-measured for smart cards," in *Proceedings of E-smart*, 2001, pp. 200–210.

[48] B. Rogers, M. Prvulovic, and Y. Solihin, "Efficient Data Protection for Distributed Shared Memory Multiprocessors," in *Proc. Int'l Conf. on Paral. Arch. and Comp. Techn. (PACT)*, Sep. 2006, pp. 84–94.

[49] W. Schindler, "A timing attack against RSA with Chinese remainder theorem," in *Proc. Crypto. HW and Emb. Sys. (CHES)*, 2000, pp. 109–124.

[50] H. Sekiguchi and S. Seto, "Study on maximum receivable distance for radiated emission of information technology equipment causing information leakage," *IEEE Trans. Electromagn. Compat.*, vol. 55, no. 3, pp. 547–554, June 2013.

[51] A. Shamir and E. Tromer, "Acoustic cryptanalysis (On nosy people and noisy machines)," http://tau.ac.il/˜tromer/acoustic/.

[52] W. Shi and H.-H. Lee, "Authentication Control Point and Its Implications for Secure Processor Design," in *Proc. 39th Int'l Symp. on Microarchitecture*, 2006.

[53] W. Shi, H.-H. Lee *et al.*, "Architectural Support for High Speed Protection of Memory Integrity and Confidentiality in Multiprocessor Systems," in *Proc. Int'l Conf. on Paral. Arch. and Comp. Techn. (PACT)*, 2004.

[54] ——, "High Efficiency Counter Mode Security Architecture via Prediction and Precomputation," in *Proc. 32nd Int'l Symp. on Comp. Arch. (ISCA)*, 2005.

[55] W. Shi, H.-H. S. Lee *et al.*, "Towards the Issues in Architectural Support for Protection of Software Execution," in *Workshop on Arch. Sup. for Sec. and Anti-virus (WASSA)*, 2004.

[56] E. Suh, C. W. O'Donnel *et al.*, "Design and Implementation of the AEGIS Single-Chip Secure Processor Using Physical Random Functions ," in *Proc. 32nd Int'l Symp. on Comp. Arch. (ISCA)*, 2005.

[57] G. Suh, D. Clarke *et al.*, "AEGIS: Architecture for Tamper-Evident and Tamper-Resistant Processing," in *Proc. 17th International Conference on Supercomputing*, 2003.

[58] G. E. Suh, D. Clarke *et al.*, "Efficient Memory Integrity Verification and Encryption for Secure Processor," in *Proc. 36th Int'l Symp. on Microarchitecture*, 2003.

[59] H. Tanaka, "Information leakage via electromagnetic emanations and evaluation of Tempest countermeasures," in *Proc. 3rd Int'l Conf.on Inf. Sys. Sec.(ICISS)*, 2007, pp. 167–179.

[60] H. Tanaka, O. Takizawa, and A. Yamamura, "Evaluation and improvement of Tempest fonts," in *Springer LNCS, Vol. 3325*, 2005, pp. 457–469.

[61] Y. Tsunoo, E. Tsujihara *et al.*, "Cryptanalysis of block ciphers implemented on computers with cache," in *Proc. Int'l Symp.on Information Theory and its Applications*, 2002, pp. 803–806.

[62] W. van Eck, "Electromagnetic radiation from video display units: an eavesdropping risk?" *Computers and Security*, pp. 269–286, Dec. 1985.

[63] Z. Wang and R. B. Lee, "New cache designs for thwarting software cache-based side channel attacks," in *Proc. 34th Int'l Symp. on Comp. Arch. (ISCA)*. ACM, 2007, pp. 494–505.

[64] C. Yan, B. Rogers *et al.*, "Improving Cost, Performance, and Security of Memory Encryption and Authentication," in *Proc. 33rd Int'l Symp. on Comp. Arch. (ISCA)*, 2006, pp. 179–190.

[65] J. Yang, Y. Zhang, and L. Gao, "Fast Secure Processor for Inhibiting Software Piracy and Tampering," in *Proc. 36th Int'l Symp. on Microarchitecture*, 2003.

[66] J. Young, "How old is Tempest?" *Online response collection, http://cryptome.org/tempest-old.htm*, 2000.

[67] A. Zajic and M. Prvulovic, "Experimental demonstration of electromagnetic information leakage from modern processor-memory systems," *IEEE Trans. Electromagn. Compat.*, vol. 99, no. 3, pp. 1–9, 2014.

[68] Y. Zhang, L. Gao *et al.*, "SENSS: Security Enhancement to Symmetric Shared Memory Multiprocessors," in *Proc. 11th Int'l Symp. on High-Perf. Comp. Arch. (HPCA)*, 2005.